Mutual Information Preserving Analysis of a Single Layer Feedforward Network

Alireza M. Javid, Saikat Chatterjee, Mikael Skoglund
Department of Information Science and Engineering
School of Electrical Engg. and Computer Sc., KTH Royal Institute of Technology, Stockholm, Sweden
{almj, sach, skoglund}@kth.se

***Abstract*—We construct a single layer feed forward network and analyze the constructed system using information theoretic tools, such as mutual information and data processing inequality. We derive a threshold on the number of hidden nodes required to achieve a good classification performance. Classification performance is expected to saturate as we increase the number of hidden nodes more than the threshold. The threshold is further verified by experimental studies on benchmark datasets.**

***Index Terms*—Neural networks, mutual information, extreme learning machine, invertible function.**

I. Introduction

A feed-forward artificial neural network (ANN) is comprised of multiple blocks where each block has a linear transform followed by a non-linear transform. Output of ANN can be interpreted as a feature representation of an input signal to the ANN, and the feature is projected to achieve a target. Typically, linear projection is used to reach the target. A single layer feedforward neural network (SLFN) can be regarded as one such (fundamental) block in a multi-layer ANN. In this article, our interest is a theoretical analysis of SLFN, regarding its number of hidden nodes.

In SLFN, we have a linear transform of the input signal followed by a non-linear transform and then, a linear transform to produce a target output. Let us denote the input vector and target vector of the SLFN by $\mathbf{x} \in \mathbb{R}^P$ and $\mathbf{t} \in \mathbb{R}^Q$, respectively. In our case, $\mathbf{x}$ is concatenation of the input signal and a scalar value 'one' to realize the effect of 'bias' in SLFN. In this way, the signal transformation in the SLFN to create the feature vector $\mathbf{y} \in \mathbb{R}^n$ is

$$\mathbf{y} = \mathbf{f}(\mathbf{x}) = \mathbf{g}(\mathbf{W}\mathbf{x}), \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{n \times P}$ is called input linear transform (or weight matrix) and $\mathbf{g} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is a non-linear transform. The non-linear transform $\mathbf{g}$ is comprised of element-wise non-linear transforms, that means scalar-wise non-linear function is used. An element-wise non-linear transform is called activation function. An activation function unit is also called a hidden node in ANN community. A preferred activation function, highly used today, is rectified linear unit (ReLU). ReLU has useful properties for training ANN, for example its gradient is a step function and it is a scale invariant function. Output of the SLFN is

$$\tilde{\mathbf{t}} = \mathbf{O}\mathbf{y} = \mathbf{O}\mathbf{g}(\mathbf{W}\mathbf{x}), \tag{2}$$

where $\mathbf{O} \in \mathbb{R}^{Q \times n}$ is called output matrix. Using appropriate cost functions, such as mean square error or cross-entropy, a standard practice is to optimize the parameters $\mathbf{O}$ and $\mathbf{W}$. SLFN is much used in classification tasks [1], [2].

An interesting question is how many number of hidden nodes need to be chosen for SLFN? The number of hidden nodes determines size of SLFN. Prevalent practice to choose this number is via experimental studies. It is known that power of ANN, and even SLFN, lies in producing informative non-linear feature vector $\mathbf{y}$ for a classification task. Note that the dimension of feature vector $\mathbf{y}$ is equal to the number of hidden nodes in SLFN. As the number of hidden nodes increases, sizes of $\mathbf{O}$ and $\mathbf{W}$ matrices increase – leading to higher model complexity and curse of dimensionality. Naturally, a logic can be that we should refrain from using a high number of hidden nodes in SLFN. In practice, an interesting behavior happens. There are many experimental studies where it has been observed that when $\mathbf{W}$ is chosen as an instance of random matrix and number of hidden nodes is increased, the classification performance improves initially and then saturates [3], [4]. This random instance based SLFN is called extreme learning machine (ELM). Performance rarely degrades with increase in number of nodes n for ELM. Our own experimental experience also corroborates the same. This behavior of ELM is counter-intuitive to the natural logic. Our main interest in this article is to provide a theoretical explanation of this behavior using tools from information theory [5].

A. Our contribution

Using mutual information-based analysis and data processing inequality, we endeavor to explain the behavior. In pursuit of theoretical explanation, we first design a new SLFN model where input matrix $\mathbf{W}$ is constructed as a product of two matrices, and ReLU is used as the activation function. We introduce a dimensionless quantity called 'node-to-input-dimension-ratio', denoted by ρ, as follows

$$\rho = \frac{\text{number of hidden nodes}}{\text{dimension of input vector}} = \frac{n}{P}. \tag{3}$$

We show that when ρ exceeds a threshold, then the constructed SLFN preserves mutual information between input signal $\mathbf{x}$ and the target vector $\mathbf{t}$ via the signal transformation in generating feature vector $\mathbf{y}$. Our hypothesis is that when we have preservation of mutual information, we will observe a

978-1-5386-5005-9/18/$31.00 ©2018 IEEE

saturation in classification performance. Before the preservation of mutual information is achieved, we will witness a high rate in improvement for classification performance. We show that the threshold value of ρ is equal to two when we achieve preservation of mutual information. That means, the constructed SLFN with number of hidden nodes twice than the input vector dimension will start to show a saturation trend in classification performance. Simulation experiments using six datasets for image classification tasks corroborate our hypothesis that the saturation trend becomes visible when ρ exceeds the threshold value two.

B. Relevant literature

Connections between Information theory, inference and learning have been investigated in literature [6]. While there exists several measures that can be used for feature selection in the literature, mutual information based feature selection has been studied significantly [7]. One of the pioneer work in this regard is [8] where a greedy feature selection algorithm was devised which employs both the mutual information with respect to the output target and with respect to the previously-selected features. Commonly, either a linear feature extractor is followed by a nonlinear classifier or a nonlinear feature is employed for a linear classification. Mutual information based design for linear feature extraction was carried out in [9] where a component-by-component gradient-ascent is used to maximize the mutual information between the corresponding feature vector and class target label. As for nonlinear feature selection, Bennasar et. al. [10] introduced joint mutual information maximization which helps to avoid selection of redundant and irrelevant features. In the case of neural networks which can be seen as a nonlinear feature generator, there exits several efforts on information-based design of network in order to ensure better generalization performance [11]–[14].

II. Mutual Information-based Analysis

In a classification task, we use the Q-dimensional target vector $\mathbf{t}$ as a discrete variable with indexed representation of 1-out-of-Q-classes. A target variable (vector) instance has only one scalar component that is one, and the other scalar components are zeros. Mutual information (MI) between the input signal vector $\mathbf{x}$ and the target vector $\mathbf{t}$ is given by

$$I(\mathbf{x}, \mathbf{t}) = h(\mathbf{x}) - h(\mathbf{x}|\mathbf{t}), \tag{4}$$

where $h(.)$ is the differential entropy, defined as

$$h(\mathbf{x}) = -\int p(\mathbf{x}) \, \log p(\mathbf{x}) \, d\mathbf{x}. \tag{5}$$

The MI measures the information content about one variable given another variable. In other words, MI measures the reduction of uncertainty of a variable when another variable is known. Intuitively, features with a high quantity of MI with respect to target class is more suitable to use for a classification task. According to data processing inequality, the following result holds for any deterministic transformation $\mathbf{h}(\mathbf{x})$:

$$I(\mathbf{h}(\mathbf{x}), \mathbf{t}) \leq I(\mathbf{x}, \mathbf{t}). \tag{6}$$

The equality in the above relation only holds when $\mathbf{h}(\mathbf{x})$ is invertible or leads to a sufficient statistics with respect to $\mathbf{t}$ [5]. Note that any transform $\mathbf{h}(\mathbf{x})$ can not improve the MI between $\mathbf{x}$ and $\mathbf{t}$. A transform $\mathbf{h}(\mathbf{x})$ can be interpreted as a feature vector that a neural network provides for a classification task. It is beneficial to work with a feature vector $\mathbf{h}(\mathbf{x})$ such that we have highest possible MI, that means we work with the equality relation $I(\mathbf{h}(\mathbf{x}), \mathbf{t}) = I(\mathbf{x}, \mathbf{t})$.

A. Proposed SLFN Architecture

In the proposed SLFN, we construct the input matrix $\mathbf{W} \in \mathbb{R}^{n \times P}$ as a product of two matrices

$$\mathbf{W} \triangleq \mathbf{V}\mathbf{A} \tag{7}$$

where $\mathbf{A} \in \mathbb{R}^{m \times P}$ and $\mathbf{V}$ has a special deterministic structure as follows

$$\mathbf{V} = \begin{bmatrix} \mathbf{I}_m \\ -\mathbf{I}_m \end{bmatrix}. \tag{8}$$

Here $\mathbf{I}_m$ is m-dimensional identity matrix and $\mathbf{V}$ is a $n \times m$-dimensional matrix; note that $n = 2m$. In case of SLFN and recalling (1), we have $\mathbf{h}(\mathbf{x}) = \mathbf{f}(\mathbf{x}) = \mathbf{y} = \mathbf{g}(\mathbf{W}\mathbf{x})$. For ELM, note that we choose $\mathbf{W}$ as an instance of random matrix and we do not optimize $\mathbf{W}$. We use ReLU activation function to construct the non-linear transform $\mathbf{g}(.)$. In our proposed SLFN, we use scalar-wise ReLU activation units. ReLU function is

$$g(x) = \max(0, x) = \begin{cases} x & \text{if } x \geq 0, \\ 0 & \text{if } x < 0. \end{cases} \tag{9}$$

Let us denote the j'th data-and-target pair in a training dataset as $(\mathbf{x}^{(j)}, \mathbf{t}^{(j)})$. We assume that there are J such pairs in the training dataset. Following (2), the output of proposed SLFN as the target prediction for input data $\mathbf{x}^{(j)}$ is $\tilde{\mathbf{t}}^{(j)} = \mathbf{O}\mathbf{y}^{(j)} = \mathbf{O}\mathbf{g}(\mathbf{W}\mathbf{x}^{(j)}) = \mathbf{O}\mathbf{g}(\mathbf{V}\mathbf{A}\mathbf{x}^{(j)})$. Then, using mean square error (MSE) as an example cost function and considering the notion of MI, a relevant optimization problem can be

$$\min_{\mathbf{O},\mathbf{A}} \frac{1}{J} \sum_j \|\mathbf{t}^{(j)} - \tilde{\mathbf{t}}^{(j)}\|^2 \text{ s.t. } \begin{cases} I(\mathbf{y}, \mathbf{t}) = I(\mathbf{x}, \mathbf{t}), \\ \|\mathbf{O}\|^2 \leq \epsilon_1, \|\mathbf{A}\|^2 \leq \epsilon_2. \end{cases} \tag{10}$$

The first constraint $I(\mathbf{y}, \mathbf{t}) = I(\mathbf{x}, \mathbf{t})$ is the mutual information preserving constraint. The second condition $\|\mathbf{O}\|^2 \leq \epsilon_1, \|\mathbf{A}\|^2 \leq \epsilon_2$ is to regularize parameters. In other words, considering invertibility, if we assume that there exists a function $\mathbf{k} : \mathbb{R}^n \to \mathbb{R}^P$ such that $\mathbf{k}(\mathbf{y}) = \mathbf{x}$, then the optimization problem (10) is equivalent to

$$\min_{\mathbf{O},\mathbf{A}} \frac{1}{J} \sum_j \|\mathbf{t}^{(j)} - \tilde{\mathbf{t}}^{(j)}\|^2 \text{ s.t. } \begin{cases} \mathbf{k}(\mathbf{y}) = \mathbf{x}, \\ \|\mathbf{O}\|^2 \leq \epsilon_1, \|\mathbf{A}\|^2 \leq \epsilon_2. \end{cases} \tag{11}$$

In our proposed SLFN, we choose $\mathbf{A}$ matrix as an instance of a random matrix, for example, elements of $\mathbf{A}$ matrix is drawn from iid Gaussian or uniform distribution. Once the $\mathbf{A}$ matrix is chosen it remains fixed for training and testing. This choice is motivated by ELM construction. Therefore, we only optimize $\mathbf{O}$ matrix in training as follows

$$\min_{\mathbf{O}} \frac{1}{J} \sum_j \|\mathbf{t}^{(j)} - \tilde{\mathbf{t}}^{(j)}\|^2 \text{ s.t. } \begin{cases} \mathbf{k}(\mathbf{y}) = \mathbf{x}, \\ \|\mathbf{O}\|^2 \leq \epsilon_1. \end{cases} \tag{12}$$

For a chosen $\mathbf{A}$ matrix we can always compute feature vector $\mathbf{y}$ due to the generative model $\mathbf{y} = \mathbf{g}(\mathbf{VAx})$. Therefore the above optimization problem takes the following form

$$\min_{\mathbf{O}} \frac{1}{J} \sum_{j} \|\mathbf{t}^{(j)} - \mathbf{O}\mathbf{y}^{(j)}\|^2 + \lambda \|\mathbf{O}\|^2 \text{ s.t. } \mathbf{k}(\mathbf{y}) = \mathbf{x}, \quad (13)$$

where λ is a Lagrangian parameter. We show that the proposed SLFN has a suitable $\mathbf{k}(.)$ function under certain technical conditions on its architecture.

B. Conditions for Invertibility

For an input vector $\boldsymbol{\gamma} \in \mathbb{R}^m$, the non-linear function $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^n$ is a stack of $g(.)$ functions such that each scalar component of input vector $\boldsymbol{\gamma}$ is treated independently, that is a scalar-wise use of $g(.)$ function. We used ReLU function as the activation function $g(.)$. We first mention about progression property of ReLU function, shown in [15].

Property 1 (Progression Property [15]:)**.** *When $g(.)$ function is ReLU then we have the progression property (PP). There are two known linear transformations $\mathbf{V} \in \mathbb{R}^{n \times m}$ and $\mathbf{U} \in \mathbb{R}^{m \times n}$ such that $\mathbf{U}\mathbf{g}(\mathbf{V}\mathbf{z}) = \mathbf{z}, \forall \mathbf{z} \in \mathbb{R}^m$. For ReLU, $\mathbf{V}$ has the special structure shown in* (8)*, and $\mathbf{U} = [\mathbf{I}_m \ -\mathbf{I}_m]$.*

Property 2 (Rank of Random Matrix [16])**.** *A real random matrix has full rank with probability one.*

Proposition 1. *For the proposed SLFN, when $n \geq 2P$, there exists an invertible transform $\mathbf{k}(\mathbf{y}) = \mathbf{x}$.*

Proof: As $n = 2m$, for $n \geq 2P$ we have $m \geq P$. In that case, $\mathbf{A}$ is a full column rank matrix using property 2. Then $\mathbf{k}(\mathbf{y}) = \mathbf{A}^{\dagger}\mathbf{U}\mathbf{y} = \mathbf{A}^{\dagger}\mathbf{z} = \mathbf{x}$ where $\mathbf{z} \triangleq \mathbf{A}\mathbf{x}$ and $\dagger$ denotes pseudo-inverse. □

The above proposition confirms that when we choose $\mathbf{A}$ matrix as an instance of random matrix and node-to-input-dimension-ratio $\rho = \frac{n}{P} \geq 2$, we have invertible relation $\mathbf{k}(\mathbf{y}) = \mathbf{x}$, in turn preservation of mutual information. Therefore, in the region $\rho \geq 2$, the optimization problem (13) is equivalent to a standard regularized least-squares form

$$\min_{\mathbf{O}} \frac{1}{J} \sum_{j} \|\mathbf{t}^{(j)} - \mathbf{O}\mathbf{y}^{(j)}\|^2 + \lambda \|\mathbf{O}\|^2. \quad (14)$$

In our proposed SLFN, we have chosen $\mathbf{A} \in \mathbb{R}^{m \times P}$ as an instance of random matrix with $m \geq P$ that ensures full column rank of $\mathbf{A}$ matrix. A natural question is what happens if we wish to optimize $\mathbf{A}$, as shown in (11). In that case we need to ensure an optimized $\mathbf{A}$ matrix with full column rank to achieve invertibility and the optimization problem (11) is equivalent to

$$\min_{\mathbf{O},\mathbf{A}} \frac{1}{J} \sum_{j} \|\mathbf{t}^{(j)} - \tilde{\mathbf{t}}^{(j)}\|^2 \text{ s.t. } \begin{cases} \text{rank}(\mathbf{A}) = P, \\ \|\mathbf{O}\|^2 \leq \epsilon_1, \|\mathbf{A}\|^2 \leq \epsilon_2. \end{cases} \quad (15)$$

In this case, our educated guess is that we start with random initialization of $\mathbf{A} \in \mathbb{R}^{m \times P}$ matrix where $m \geq P$, and use gradient search based techniques (such as back propagation algorithm) to optimize a relevant cost function without explicit requirement of full column rank. In practice, the optimized $\mathbf{A}$ matrix will turn out to be a full rank matrix. A relevant cost function can be

$$\min_{\mathbf{O},\mathbf{A}} \frac{1}{J} \sum_{j} \|\mathbf{t}^{(j)} - \mathbf{O}\mathbf{g}(\mathbf{V}\mathbf{A}\mathbf{x}^{(j)})\|^2 + \lambda_1 \|\mathbf{O}\|^2 + \lambda_2 \|\mathbf{A}\|^2. \quad (16)$$

In case we do not have a full rank matrix, we can use a small perturbation to the $\mathbf{A}$ matrix to make it full rank. This is a standard signal processing trick.

TABLE I: Databases for multi-class classification

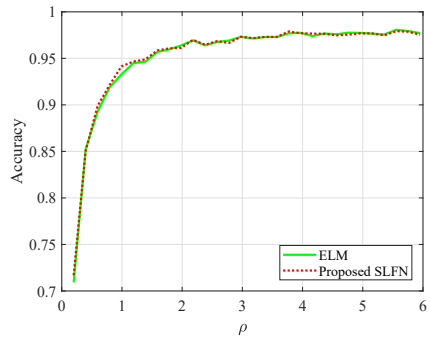
Database	# of train data	# of test data	Input dimension (P)	# of classes (Q)	Random Partition
Extended YaleB	1600	800	504	38	Yes
AR	1800	800	540	100	Yes
Scene15	3000	1400	3000	15	Yes
Caltech101	6000	3000	3000	102	Yes
NORB	24300	24300	2048	5	No
MNIST	60000	10000	784	10	No

III. Experimental Results

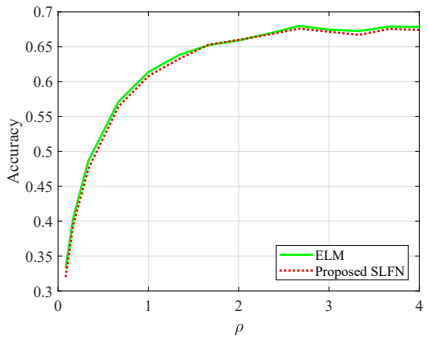
Our hypothesis is that classification performance of proposed SLFN improves on a high rate when we do not have preservation of mutual information, that means in the region $\rho < 2$. For $\rho \geq 2$ we have preservation of mutual information and we will observe a saturation in classification performance. In order to verify our theoretical arguments, we perform simulation experiments on a number of benchmark classification databases. We evaluate the test classification accuracy of proposed SLFN while ρ is increased, that means the number of hidden nodes is increased. We also show performance of ELM where there is no clear theoretical argument on the choice of number of hidden nodes.

The characteristics of the databases used in the experiment is shown in Table I. These are standard databases in image classification tasks. Each database is partitioned into two disjoint sets of training and testing. The term 'Random Partition' in Table I determines if a partition of corresponding database is performed randomly or not. In case of NORB and MNIST databases, the training and testing sets are predetermined by the designers of the databases and hence the testing and training sets are fixed for those two databases. Note that we have chosen the databases in which the input dimension P is large enough so that the performance improvement in the range of $1 < \rho < 2$ be tangible.

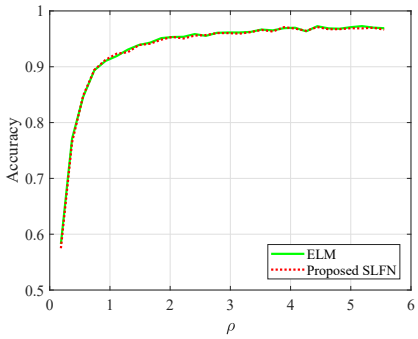
The simulation results are shown in Figure 1. We show the average classification accuracy over 10 trials of the proposed SLFN and ELM where regularized least-squares optimization was carried out. For the case of proposed SLFN, it can be seen that the performance improvement in classification accuracy is tangible when $\rho < 2$, but it slowly saturates when ρ goes larger than two. This experimental result verifies our hypothesis. Note than although the convergence happens in all cases but the rate of convergence is different for each database. For instance, in Scene15 database, the convergence speed is higher compared to the other cases, and for MNIST



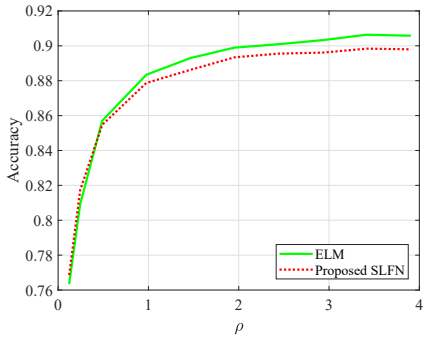
(a) For ExtendedYaleB database



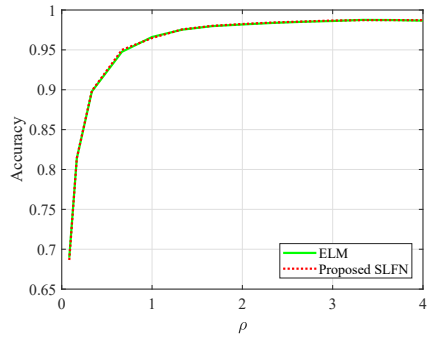
(d) For Caltech101 database



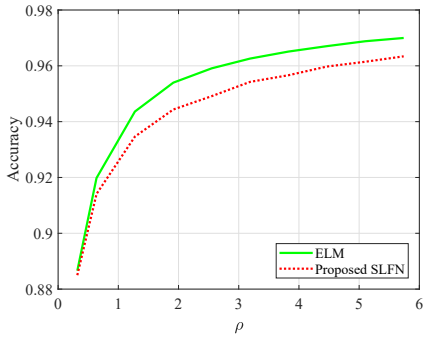
(b) For AR database



(e) For NORB database



(c) For Scene15 database



(f) For MNIST database

Fig. 1: Test classification accuracy of ELM and the proposed SLFN on different benchmark databases versus the node-to-input-dimension ratio ρ. (a) For ExtendedYaleB database, $\lambda = 10^7$. (b) For AR database, $\lambda = 10^8$. (c) For Scene15 database, $\lambda = 1$. (d) For Caltech101 databse, $\lambda = 10^2$. (e) For NORB databse, $\lambda = 10^3$. (f) For MNIST database, $\lambda = 10^{-1}$.

database it is the slowest one. The relationship between this convergence rate and the characteristics of each database is not clear to us yet, but as a trend, we can see than the convergence would be slower as the number of training and testing samples in a dataset increases. Another important point is that ELM performance is similar like the proposed SLFN. We have already mentioned that there is no theoretical argument on the choice of hidden nodes for ELM. Therefore, considering our theoretical reasoning, ELM also is able to preserve mutual information, in turn invertibility when $\rho \geq 2$. There can be a soft argument to justify the behavior of ELM. When $n \geq 2P$, we have $\mathbf{y}$ vector that has around half of the elements which are non-zeros. If this holds for most of the trials then it is possible to reconstruct $\mathbf{x}$ from $\mathbf{y}$ by constructing appropriate linear set of equations. We have to only consider positive elements of $\mathbf{y}$ for this purpose. This is a supporting argument for ELM, but not sufficiently justified in theory.

IV. Conclusions

We conclude that it is possible to design a single layer feedforward network architecture that can be analyzed to determine how many hidden nodes are required to achieve a good performance. It is possible to use existing knowledge from the vast area of signals and systems, such as information theoretic tools for analysis. A prime question arises and it remains unanswered: why extreme learning machine and our constructed SLFN provides similar performance?

REFERENCES

[1] G.-B. Huang, Y.-Q. Chen, and H. A. Babri, "Classification ability of single hidden layer feedforward neural networks," *IEEE Transactions on Neural Networks*, vol. 11, pp. 799–801, May 2000.

[2] V. P. Vishwakarma and M. N. Gupta, "A new learning algorithm for single hidden layer feedforward neural networks," *International Journal of Computer Applications*, 2011.

[3] "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Transactions on Neural Networks*, vol. 17, pp. 879–892, July 2006.

[4] G.-B. Huang, "What are extreme learning machines? filling the gap between frank rosenblatt's dream and john von neumann's puzzle," *Cognitive Computation*, vol. 7, pp. 263–278, Jun 2015.

[5] T. Cover and J. Thomas, *Elements of Information Theory, 2nd Edition*. Wiley, 2006.

[6] D. MacKay, *Information Theory, Inference and Learning Algorithms*. Cambridge Univ. Press, 2003.

[7] J. R. Vergara and P. A. Estévez, "A review of feature selection methods based on mutual information," *Neural Computing and Applications*, vol. 24, pp. 175–186, Jan 2014.

[8] R. Battiti, "Using mutual information for selecting features in supervised neural net learning," *IEEE Transactions on Neural Networks*, vol. 5, pp. 537–550, Jul 1994.

[9] J. M. Leiva-Murillo and A. Artes-Rodriguez, "Maximization of mutual information for supervised linear feature extraction," *IEEE Transactions on Neural Networks*, vol. 18, no. 5, pp. 1433–1441, 2007.

[10] M. Bennasar, Y. Hicks, and R. Setchi, "Feature selection using joint mutual information maximisation," *Expert Systems with Applications*, vol. 42, no. 22, pp. 8520 – 8532, 2015.

[11] T. T. Nguyen and J. Choi, "Layer-wise learning of stochastic neural networks with information bottleneck," *https://arxiv.org/abs/1712.01272*, 2017.

[12] R. A. Amjad and B. C. Geiger, "How (not) to train your neural network using the information bottleneck principle," *https://arxiv.org/pdf/1802.09766*, 2018.

[13] X. Yao and Y. Liu, "Evolving neural network ensembles by minimization of mutual information," *Int. J. Hybrid Intell. Syst.*, vol. 1, pp. 12–21, Apr. 2004.

[14] S. Yu and J. C. Principe, "Understanding autoencoders with information theoretic concepts," *https://arxiv.org/abs/1804.00057*, 2018.

[15] S. Chatterjee, A. M. Javid, M. Sadeghi, P. P. Mitra, and M. Skoglund, "Progressive learning for systematic design of large neural networks," *https://arxiv.org/abs/1710.08177*, 2017.

[16] X. Feng and Z. Zhang, "The rank of a random matrix," *Applied Mathematics and Computation*, vol. 185, no. 1, pp. 689 – 694, 2007.